

**REMARKS**

Applicant respectfully requests reconsideration and allowance of the subject application. Claims 1-27 are pending in this application.

**35 U.S.C. § 102**

Claims 4-11 stand rejected under 35 U.S.C. §102(e) as being unpatentable over .NET Framework Essentials by Lam (hereinafter “Lam”). Applicant respectfully submits that claims 4-11 are not anticipated by Lam.

The subject matter of claims 4-11 was conceived prior to the June 2001 publication date of Lam, and filing of the above identified application was diligently pursued from prior to June 2001 until the filing of the above identified application on July 10, 2001. Accompanying this response is a Declaration under 37 C.F.R. §1.131 with supporting documentation to evidence that the invention was conceived in the United States prior to the June 2001 publication date of Lam, coupled with due diligence from prior to the June 2001 publication date of Lam to filing of the application. Accordingly, the Lam reference should be removed as a reference because it is not prior art.

Accordingly, Applicant respectfully requests that the rejection of claims 4-11 over Lam be withdrawn.

Claim 12 stands rejected under 35 U.S.C. §102(e) as being unpatentable over US Publication No. 2002/0169679 to Neumayer (hereinafter “Neumayer”). Applicant respectfully submits that claim 12 is not anticipated by Neumayer.

Neumayer is directed to an aggregation engine for an electronic commerce system (see, Title). As discussed in the abstract of Neumayer, Neumayer

describes an aggregation engine for use in electronic commerce systems, such as an enterprise procurement system or an electronic marketplace, that automatically aggregates buyer demands according to an aggregation rule so as to enable the creation of fewer purchase orders and to take advantage of bulk buying power.

Claim 12 recites:

An XmlValidatingReader class of an application program interface, embodied on one or more computer readable media, that enables DTD, XDR and XSD schema validation, the XmlValidatingReader class comprising:

a ValidationType property that enables obtaining an indication of what type of validation to perform on a document;

a Read method that enables reading of nodes of the document so that validation of the document can be performed.

Applicant respectfully submits that Neumayer does not disclose the XmlValidatingReader class of an application program interface of claim 12.

It appears that the XMLValidator class of Neumayer is being relied on as disclosing the XmlValidatingReader class of claim 12 (see, May 26 Office Action at ¶8, p. 9). However, there is no discussion or mention in Neumayer of the XMLValidator including a ValidationType property that enables obtaining an indication of what type of validation to perform on a document.

In the May 26 Office Action at ¶8, pp. 9-10, paragraph 33 at pp. 3-4 of Neumayer is cited as disclosing the ValidationType property. However, this portion of Neumayer reads as follows:

[0033] In step 160, the incoming data is validated. This could be accomplished through a class called XMLValidator, which is a helper class to check if the XML data is valid. A method of validateXML can be used to check the given XML against the schema.

There is no discussion or mention in this cited portion of Neumayer of a ValidationType property as recited in claim 12. Although this cited portion discusses the XMLValidator class, and that a validateXML method can be used to check the given XML against the schema, nowhere is there any mention of a ValidationType property. The mere disclosure of a method checking the given XML against the schema does not disclose a property that enables obtaining an indication of what type of validation to perform on a document. There is no discussion or mention in this cited portion of Neumayer, or elsewhere in Neumayer, of different types of validation that may be performed on a document, much less of a property of the XMLValidator class of Neumayer that enables obtaining an indication of what type of validation to perform on a document. As there is no discussion or mention of a property of the XMLValidator class of Neumayer that enables obtaining an indication of what type of validation to perform on a document in Neumayer, Applicant respectfully submits that Neumayer cannot disclose an XmlValidatingReader class comprising a ValidationType property that enables obtaining an indication of what type of validation to perform on a document as recited in claim 12.

It was further asserted in the May 26 Office Action at ¶13, p. 26 that:

For the ValidationType property that enables obtaining and indication of what type of validation to perform on a document, Neumayer system does provide validation on a XML data, then creates a rule object for further processing where rule class has a RuleID which indicates the properties of the object (see page 3, paragraphs 34-36).

However, the cited portions of Neumayer (paragraphs 34-36) discuss step 170 of Fig. 2 where the incoming XML document is processed to extract the demands to

be aggregated and the aggregation rule. An aggregation rule is a set of criteria used to group demands together (see, p. 2, ¶ 27). Thus, the processing discussed in step 170 is for aggregating demands together, not performing validation on a document. Furthermore, this processing in step 170 is performed after the validation occurs in Neumayer (the validation of Neumayer occurring in step 160). As this processing of the XML document to aggregate demands together in Neumayer is separate from, and performed after, the validation of Neumayer, Applicant respectfully submits that this processing does not disclose any validation. As such, Applicant respectfully submits that this processing of step 170 cannot disclose an XmlValidatingReader class comprising a ValidationType property that enables obtaining an indication of what type of validation to perform on a document as recited in claim 12.

For at least these reasons, Applicant respectfully submits that claim 12 is allowable over Neumayer.

Applicant respectfully requests that the §102 rejections be withdrawn.

### **35 U.S.C. § 103**

Claims 1, 16, 19, 22, and 25 stand rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,418,448 to Sarkar (hereinafter “Sarkar”) in view of U.S. Patent No. 6,209,124 to Vermeire et al. (hereinafter “Vermeire”) and further in view of U.S. Patent No. 6,349,343 to Foody et al. (hereinafter “Foody”). Applicant respectfully submits that claims 1, 16, 19, 22, and 25 are not obvious over Sarkar in view of Vermeire and further in view of Foody.

As discussed in the abstract of Sarkar, Sarkar is directed to a system for navigation through multiple documents in Extensible Markup Language (XML) and Resource Description Framework (RDF) to inspect data/metadata in order to either start a transaction on selected item(s) in separate thin client window(s) with persistent connectivity through Internet Inter ORB Protocol or implicitly trigger read-only queries in Structured Query Language (SQL) represented in RDF against a unified virtual Database defined over multiple physical disparate object relational databases over the web. An implicitly generated query retrieves desired sets of properties and entities presented in documents of XML and RDF for further navigation. Container types in RDF are mapped by Sarkar to record and table types in a normalized relational model where URIs locating elements in relational schema components over the web are stored as primary keys/foreign keys in normalized tables. Methods and operators on such web objects are defined as part of user-defined package definitions in object relational schema where object request brokers apply such methods or operators on result sets from relational operations anywhere on the web.

Vermeire is directed to a method of markup language accessing of host systems and data using a constructed intermediary (see, Title). In Vermeire, information about host computer software application structures, called metadata, is blended with either the XML representation or the binary data values to generate binary data for use with a host system or an XML representation for use in mark-up language applications (see, col. 5, lines 2-6). A constructed intermediary is utilized which is user defined based upon the application language utilized by the host computer (see, col. 5, lines 6-9). The intermediary functions to

restructure in-memory binary data streams received from the host into XML documents and to restructure XML documents into binary data streams capable of acting with the host machine and its program applications (see, col. 5, lines 12-16).

Foody is directed to a single system in a digital computer enabling software objects from multiple heterogeneous object systems to interoperate bi-directionally and to be combined in the creation of a larger software system (see, col. 6, lines 52-56). As discussed in the abstract of Foody, objects from a foreign object system are unmodified, yet appear to be native to the object system in which they are used or accessed. A native proxy object (indistinguishable from other native objects) is constructed for the real foreign object. The proxy object contains an identifier to the real object, as well as a pointer to a software description of how to access and manipulate the object--e.g. how to call its methods, set its properties, and handle exceptions. When the proxy object is manipulated, it follows the instructions in the software description which, in turn, results in the corresponding manipulation of the foreign object.

In contrast, claim 1 recites, in part:

an application program interface to present functions used by the application to access network and computing resources of the distributed computing system, wherein the application program interface comprises a set of classes that make available standards-based support for processing XML documents, wherein the set of classes are grouped in the application program interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second of the plurality of namespaces contains an XPath parser and evaluation engine.

Applicant respectfully submits that no such application program interface is disclosed or suggested in Sarkar in view of Vermeire and further in view of Foody.

In the May 26 Office Action at ¶10, p. 12, it was asserted that:

However, Foody discloses . . . wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second of the plurality of namespaces contains an Xpath parser and evaluation engine (col. 11, lines 53-67 "...information... constructed... as NameSpaces (XSLT would be inherent in this process since it consist of implementing XML) are enumerated... would typically use subclasses of each... classes... set information..." also see FIG. 2b and related discussion.

This cited portion of Foody (col. 11, lines 53-67) reads as follows:

The above information is typically constructed by the OSA as NameSpaces are enumerated, however, it may be constructed directly by the user of the system. If the OSA doesn't override any capabilities, as it enumerates its contents, it would typically use subclasses of each of the above classes which, in addition to having methods to retrieve information, have corresponding methods to set the information. Thus, as the OSA determined the contents of the object system, it would construct the necessary instances of the above classes, and call their methods to "connect" them. For example, it would construct the subclass of VClassData which supported these methods, then construct VPropData corresponding to the class' properties, then call a method to set the list of properties in the (subclass of) VClassData. FIG. 4 shows the results of an OSA constructing . . . .

The "above information" in this cited portion apparently refers to, as described in the paragraphs preceding this cited portion, the Class Description Framework consisting of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions (see, col. 11, lines 2-5). The corresponding classes used are entitled, respectively: VClassData,

VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData (see, col. 11, lines 5-8). These classes are described in additional detail in the paragraphs spanning col. 11, lines 15-52 of Foody.

This cited portion of Foody discusses constructing the necessary instances of these classes and calling their methods to connect them. Nowhere in this cited portion is there any discussion or mention of grouping a set of classes in an application program interface with a first of the plurality of namespaces containing classes and enumerations to support XSLT, and a second of the plurality of namespaces containing an XPath parser and evaluation engine. XSLT, as well as an XPath parser and evaluation engine, are not even mentioned in this cited portion. Without any discussion or mention of XSLT and an XPath parser and evaluation engine, Applicant respectfully submits that Foody cannot disclose or suggest wherein the set of classes are grouped in the application program interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second of the plurality of namespaces contains an XPath parser and evaluation engine as recited in claim 1.

Furthermore, although Applicant submits that XSLT is not inherent in the process discussed in the cited portion of Foody, assuming for the sake of argument that XSLT were inherent in this process, there still is no discussion or mention in this cited portion of the set of classes being grouped into a plurality of namespaces, one of which contains classes and enumerations to support XSLT and the other of which contains an XPath parser and evaluation engine. The mere mention of classes, and any XSLT inherent in the described process of Foody,



does not provide any disclosure or suggestion to group classes into a plurality of namespaces including the two namespaces recited in claim 1.

Accordingly, Applicant respectfully submits that Foody does not disclose or suggest wherein the set of classes are grouped in the application program interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second of the plurality of namespaces contains an XPath parser and evaluation engine as recited in claim 1. With respect to Sarkar and Vermeire, Sarkar and Vermeire are not cited as curing, and do not cure, the deficiencies of Foody discussed above.

For at least these reasons, Applicant respectfully submits that claim 1 is allowable over Sarkar in view of Vermeire and further in view of Foody.

With respect to claim 16, claim 16 recites in part:

an application programming interface to interface the one or more applications with the networking platform, wherein the application program interface comprises a set of classes that make available standards-based support for processing documents written in a markup language, wherein the set of classes are grouped in the application programming interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), a second of the plurality of namespaces contains an XPath parser and evaluation engine, and a third of the plurality of namespaces contains classes used to serialize objects into XML format documents or streams.

Applicant respectfully submits that no such application programming interface is disclosed or suggested in Sarkar in view of Vermeire and further in view of Foody.

In the May 26 Office Action at ¶10, pp. 14-15, Foody is relied on as disclosing wherein the set of classes are grouped in the application programming interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), a second of the plurality of namespaces contains an XPath parser and evaluation engine, and a third of the plurality of namespaces contains classes used to serialize objects into XML format documents or streams as recited in claim 16. Applicant respectfully disagrees and submits that these elements are not disclosed or suggested by Foody.

This cited portion of Foody (col. 11, lines 53-67) reads as follows:

The above information is typically constructed by the OSA as NameSpaces are enumerated, however, it may be constructed directly by the user of the system. If the OSA doesn't override any capabilities, as it enumerates its contents, it would typically use subclasses of each of the above classes which, in addition to having methods to retrieve information, have corresponding methods to set the information. Thus, as the OSA determined the contents of the object system, it would construct the necessary instances of the above classes, and call their methods to "connect" them. For example, it would construct the subclass of VClassData which supported these methods, then construct VPropData corresponding to the class' properties, then call a method to set the list of properties in the (subclass of) VClassData. FIG. 4 shows the results of an OSA constructing . . . .

The "above information" in this cited portion apparently refers to, as described in the paragraphs preceding this cited portion, the Class Description Framework consisting of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions (see, col. 11, lines 2-5). The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and

VExceptionData (see, col. 11, lines 5-8). These classes are described in additional detail in the paragraphs spanning col. 11, lines 15-52 of Foody.

This cited portion discusses constructing the necessary instances of these classes and calling their methods to connect them. Nowhere in this cited portion is there any discussion or mention of grouping a set of classes in an application program interface with a first of the plurality of namespaces containing classes and enumerations to support XSLT, a second of the plurality of namespaces containing an XPath parser and evaluation engine, and a third of the plurality of namespaces containing classes used to serialize objects into XML format documents or streams. None of XSLT, an XPath parser and evaluation engine, and serializing objects into XML format documents or streams are even mentioned in this cited portion. Without any such mention of XSLT, an XPath parser and evaluation, and serializing objects into XML format documents or streams, Applicant respectfully submits that Foody cannot disclose or suggest wherein the set of classes are grouped in the application programming interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), a second of the plurality of namespaces contains an XPath parser and evaluation engine, and a third of the plurality of namespaces contains classes used to serialize objects into XML format documents or streams as recited in claim 16.

Furthermore, contrary to the assertions in the May 26 Office Action, Applicant submits that XSLT is not inherent in the process discussed in the cited portion of Foody. However, assuming for the sake of argument that XSLT were inherent in this process, there still is no discussion or mention in this cited portion

of the set of classes being grouped into a plurality of namespaces, one of which contains classes and enumerations to support XSLT, another of which contains an XPath parser and evaluation engine, and another of which contains classes used to serialize objects into XML format documents or streams. The mere mention of classes, and any XSLT inherent in the described process of Foody, does not provide any disclosure or suggestion to group classes into a plurality of namespaces including the three namespaces recited in claim 16.

Accordingly, Applicant respectfully submits that Foody does not disclose or suggest wherein the set of classes are grouped in the application programming interface into a plurality of namespaces, and wherein a first of the plurality of namespaces contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), a second of the plurality of namespaces contains an XPath parser and evaluation engine, and a third of the plurality of namespaces contains classes used to serialize objects into XML format documents or streams as recited in claim 16. With respect to Sarkar and Vermeire, Sarkar and Vermeire are not cited as curing, and do not cure, the deficiencies of Foody discussed above.

For at least these reasons, Applicant respectfully submits that claim 16 is allowable over Sarkar in view of Vermeire and further in view of Foody.

With respect to claim 19, claim 19 recites:

A computer system including one or more microprocessors and one or more software programs, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services that make available support for processing XML documents, the separate commands being grouped into different namespaces including a first namespace

to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace to serialize objects into XML format documents or streams.

Applicant respectfully submits that no such one or more software programs utilizing such an application program interface is disclosed or suggested in Sarkar in view of Vermeire and further in view of Foody.

In the May 26 Office Action at ¶10, p. 17, Foody is relied on as disclosing the separate commands being grouped into different namespaces including a first namespace to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace to serialize objects into XML format documents or streams as recited in claim 19. Applicant respectfully disagrees and submits that these elements are not disclosed or suggested by Foody.

The first cited portion of Foody (col. 10-11, lines, 65-66 and 1-14) reads as follows:

Contained in Adapter and View NameSpaces is the information to describe classes. The Class Description Framework is provided to enable this capability, and to enable OSAs to override built-in functionality. The Class Description Framework consists of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions. The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData. An additional framework to describe types, the Type Description Framework as described below, is also utilized. Each of the above classes can be asked for the name of the object it represents, its type, its owner (e.g. if it describes a class property, its owner would be the class), the object system that manages it, as well as "usage codes" (described below).

The second cited portion of Foody (col. 11, lines 53-67) reads as follows:

The above information is typically constructed by the OSA as NameSpaces are enumerated, however, it may be constructed directly by the user of the system. If the OSA doesn't override any capabilities, as it enumerates its contents, it would typically use

subclasses of each of the above classes which, in addition to having methods to retrieve information, have corresponding methods to set the information. Thus, as the OSA determined the contents of the object system, it would construct the necessary instances of the above classes, and call their methods to "connect" them. For example, it would construct the subclass of VClassData which supported these methods, then construct VPropData corresponding to the class' properties, then call a method to set the list of properties in the (subclass of) VClassData. FIG. 4 shows the results of an OSA constructing . . . .

The "above information" in this second cited portion apparently refers to, as described in the paragraphs preceding this second cited portion, the Class Description Framework consisting of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions (see, col. 11, lines 2-5). The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData (see, col. 11, lines 5-8). These classes are described in additional detail in the paragraphs spanning col. 11, lines 15-52 of Foody.

These cited portions discusses constructing the necessary instances of these classes and calling their methods to connect them. Nowhere in these cited portions is there any discussion or mention of grouping a set of classes in an application program interface into different namespaces with a first namespace to support XSLT, and a second namespace to serialize objects into XML format documents or streams. XSLT, as well as serializing objects into XML format documents or streams, are not even mentioned in these cited portions. Without any such mention of XSLT and serializing objects into XML format documents or streams, Applicant respectfully submits that Foody cannot disclose or suggest the separate commands being grouped into different namespaces including a first namespace to support XSLT (Extensible Stylesheet Language Transformations)

and a second namespace to serialize objects into XML format documents or streams as recited in claim 19.

Furthermore, contrary to the assertions in the May 26 Office Action, Applicant submits that XSLT is not inherent in the process discussed in these cited portions of Foody. However, assuming for the sake of argument that XSLT were inherent in this process, there still is no discussion or mention in these cited portions of the separate commands being grouped into a plurality of namespaces, one of which is to support XSLT, and another of which is to serialize objects into XML format documents or streams. The mere mention of classes, and any XSLT inherent in the described process of Foody, does not provide any disclosure or suggestion to group commands into different namespaces including the two namespaces recited in claim 19.

Accordingly, Applicant respectfully submits that Foody does not disclose or suggest the separate commands being grouped into different namespaces including a first namespace to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace to serialize objects into XML format documents or streams as recited in claim 19. With respect to Sarkar and Vermeire, Sarkar and Vermeire are not cited as curing, and do not cure, the deficiencies of Foody discussed above.

For at least these reasons, Applicant respectfully submits that claim 19 is allowable over Sarkar in view of Vermeire and further in view of Foody.

With respect to claim 22, claim 22 recites in part:

receiving one or more calls from one or more remote devices over a network, wherein the one or more calls are to one or more functions that make available support for processing XML documents, the one or more functions being grouped into a plurality

of namespaces with a first namespace containing an XPath parser and evaluation engine and a second namespace containing classes used to serialize objects into XML format documents or streams; and

Applicant respectfully submits that no such one or more functions is disclosed or suggested in Sarkar in view of Vermeire and further in view of Foody.

In the May 26 Office Action at ¶10, p. 19, Foody is relied on as disclosing the one or more functions being grouped into a plurality of namespaces with a first namespace containing an XPath parser and evaluation engine and a second namespace containing classes used to serialize objects into XML format documents or streams as recited in claim 22. Applicant respectfully disagrees and submits that these elements are not disclosed or suggested by Foody.

The first cited portion of Foody (col. 10-11, lines, 65-66 and 1-14) reads as follows:

Contained in Adapter and View NameSpaces is the information to describe classes. The Class Description Framework is provided to enable this capability, and to enable OSAs to override built-in functionality. The Class Description Framework consists of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions. The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData. An additional framework to describe types, the Type Description Framework as described below, is also utilized. Each of the above classes can be asked for the name of the object it represents, its type, its owner (e.g. if it describes a class property, its owner would be the class), the object system that manages it, as well as "usage codes" (described below).

The second cited portion of Foody (col. 11, lines 53-67) reads as follows:

The above information is typically constructed by the OSA as NameSpaces are enumerated, however, it may be constructed directly by the user of the system. If the OSA doesn't override any capabilities, as it enumerates its contents, it would typically use subclasses of each of the above classes which, in addition to having methods to retrieve information, have corresponding methods to set



the information. Thus, as the OSA determined the contents of the object system, it would construct the necessary instances of the above classes, and call their methods to "connect" them. For example, it would construct the subclass of VClassData which supported these methods, then construct VPropData corresponding to the class' properties, then call a method to set the list of properties in the (subclass of) VClassData. FIG. 4 shows the results of an OSA constructing . . . .

The "above information" in this second cited portion apparently refers to, as described in the paragraphs preceding this second cited portion, the Class Description Framework consisting of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions (see, col. 11, lines 2-5). The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData (see, col. 11, lines 5-8). These classes are described in additional detail in the paragraphs spanning col. 11, lines 15-52 of Foody.

These cited portions discusses constructing the necessary instances of these classes and calling their methods to connect them. Nowhere in these cited portions is there any discussion or mention of one or more functions being grouped into a plurality of namespaces with a first namespace containing an XPath parser and evaluation engine and a second namespace containing classes used to serialize objects into XML format documents or streams. An XPath parser and evaluation engine, as well as serializing objects into XML format documents or streams, are not even mentioned in these cited portions. Without any such mention of an XPath parser and evaluation engine, and serializing objects into XML format documents or streams, Applicant respectfully submits that Foody cannot disclose or suggest the one or more functions being grouped into a plurality of namespaces with a first namespace containing an XPath parser and evaluation

engine and a second namespace containing classes used to serialize objects into XML format documents or streams as recited in claim 22.

Furthermore, contrary to the assertions in the May 26 Office Action, Applicant submits that XSLT is not inherent in the process discussed in these cited portions of Foody. However, assuming for the sake of argument that XSLT were inherent in this process, there still is no discussion or mention in these cited portions of the one or more functions being grouped into a plurality of namespaces, one of which containing an XPath parser and evaluation engine, and another of which containing classes used to serialize objects into XML format documents or streams. The mere mention of classes, and any XSLT inherent in the described process of Foody, does not provide any disclosure or suggestion to group commands into different namespaces including the two namespaces recited in claim 22.

Accordingly, Applicant respectfully submits that Foody does not disclose or suggest the one or more functions being grouped into a plurality of namespaces with a first namespace containing an XPath parser and evaluation engine and a second namespace containing classes used to serialize objects into XML format documents or streams as recited in claim 22. With respect to Sarkar and Vermeire, Sarkar and Vermeire are not cited as curing, and do not cure, the deficiencies of Foody discussed above.

For at least these reasons, Applicant respectfully submits that claim 22 is allowable over Sarkar in view of Vermeire and further in view of Foody.

With respect to claim 25, claim 25 recites in part:

calling, to one or more remote devices over a network, one or more functions that make available support for processing XML

documents, the one or more functions being grouped into a plurality of namespaces with a first namespace containing classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace containing classes used to serialize objects into XML format documents or streams;

Applicant respectfully submits that no such calling of one or more functions is disclosed or suggested in Sarkar in view of Vermeire and further in view of Foody.

In the May 26 Office Action at ¶10, pp. 21-22, Foody is relied on as disclosing the one or more functions being grouped into a plurality of namespaces with a first namespace containing classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace containing classes used to serialize objects into XML format documents or streams as recited in claim 25. Applicant respectfully disagrees and submits that these elements are not disclosed or suggested by Foody.

The first cited portion of Foody (col. 10-11, lines, 65-66 and 1-14) reads as follows:

Contained in Adapter and View NameSpaces is the information to describe classes. The Class Description Framework is provided to enable this capability, and to enable OSAs to override built-in functionality. The Class Description Framework consists of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions. The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData. An additional framework to describe types, the Type Description Framework as described below, is also utilized. Each of the above classes can be asked for the name of the object it represents, its type, its owner (e.g. if it describes a class property, its owner would be the class), the object system that manages it, as well as "usage codes" (described below).

The second cited portion of Foody (col. 11, lines 53-67) reads as follows:

The above information is typically constructed by the OSA as NameSpaces are enumerated, however, it may be constructed directly by the user of the system. If the OSA doesn't override any capabilities, as it enumerates its contents, it would typically use subclasses of each of the above classes which, in addition to having methods to retrieve information, have corresponding methods to set the information. Thus, as the OSA determined the contents of the object system, it would construct the necessary instances of the above classes, and call their methods to "connect" them. For example, it would construct the subclass of VClassData which supported these methods, then construct VPropData corresponding to the class' properties, then call a method to set the list of properties in the (subclass of) VClassData. FIG. 4 shows the results of an OSA constructing . . . .

The "above information" in this second cited portion apparently refers to, as described in the paragraphs preceding this second cited portion, the Class Description Framework consisting of a suite of classes which describe: classes, instances, properties, functions (including methods), arguments, and exceptions (see, col. 11, lines 2-5). The corresponding classes used are entitled, respectively: VClassData, VInstanceData, VPropData, VFunctionData, VArgumentData, and VExceptionData (see, col. 11, lines 5-8). These classes are described in additional detail in the paragraphs spanning col. 11, lines 15-52 of Foody.

These cited portions discusses constructing the necessary instances of these classes and calling their methods to connect them. Nowhere in these cited portions is there any discussion or mention of one or more functions being grouped into a plurality of namespaces with a first namespace containing classes and enumerations to support XSLT and a second namespace containing classes used to serialize objects into XML format documents or streams. XSLT, as well as serializing objects into XML format documents or streams, are not even

mentioned in these cited portions. Without any such mention of XSLT, and serializing objects into XML format documents or streams, Applicant respectfully submits that Foody cannot disclose or suggest the one or more functions being grouped into a plurality of namespaces with a first namespace containing classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace containing classes used to serialize objects into XML format documents or streams as recited in claim 25.

Furthermore, contrary to the assertions in the May 26 Office Action, Applicant submits that XSLT is not inherent in the process discussed in these cited portions of Foody. However, assuming for the sake of argument that XSLT were inherent in this process, there still is no discussion or mention in these cited portions of the one or more functions being grouped into a plurality of namespaces, one of which containing classes and enumerations to support XSLT, and another of which containing classes used to serialize objects into XML format documents or streams. The mere mention of classes, and any XSLT inherent in the described process of Foody, does not provide any disclosure or suggestion to group commands into different namespaces including the two namespaces recited in claim 25.

Accordingly, Applicant respectfully submits that Foody does not disclose or suggest the one or more functions being grouped into a plurality of namespaces with a first namespace containing classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations) and a second namespace containing classes used to serialize objects into XML format documents or streams

as recited in claim 25. With respect to Sarkar and Vermeire, Sarkar and Vermeire are not cited as curing, and do not cure, the deficiencies of Foody discussed above.

For at least these reasons, Applicant respectfully submits that claim 25 is allowable over Sarkar in view of Vermeire and further in view of Foody.

Claims 2, 17, 20, 23, and 26 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Sarkar in view of Vermeire, Foody, and further in view of Lam. Applicant respectfully submits that claims 2, 17, 20, 23, and 26 are not obvious over Sarkar in view of Vermeire, Foody, and further in view of Lam.

In view of the discussion above regarding Lam, Lam is removed as prior art. Given that claims 2, 17, 20, 23, and 26 are rejected under §103 based in part on Lam, and no art rejection is made of any of the claims using any reference or combination of references that does not include Lam, Applicant respectfully submits that the cited references, without Lam, do not disclose or suggest claims 2, 17, 20, 23, and 26.

Claims 3, 13-15, 18, 21, 24, and 27 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Sarkar in view of Vermeire, Foody, Lam and further in view of Neumayer. Applicant respectfully submits that claims 3, 13-15, 18, 21, 24, and 27 are not obvious over Sarkar in view of Vermeire, Foody, Lam and further in view of Neumayer.

In view of the discussion above regarding Lam, Lam is removed as prior art. Given that claims 3, 13-15, 18, 21, 24, and 27 are rejected under §103 based in part on Lam, and no art rejection is made of any of the claims using any reference or combination of references that does not include Lam, Applicant

respectfully submits that the cited references, without Lam, do not disclose or suggest claims 3, 13-15, 18, 21, 24, and 27.

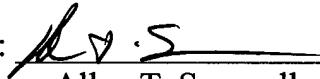
Applicant respectfully requests that the §103 rejections be withdrawn.

**Conclusion**

Claims 1-27 are in condition for allowance. Applicant respectfully requests reconsideration and issuance of the subject application. Should any matter in this case remain unresolved, the undersigned attorney respectfully requests a telephone conference with the Examiner to resolve any such outstanding matter.

Respectfully Submitted,

Date: 9/26/05

By:   
Allan T. Sponseller  
Reg. No. 38,318  
(509) 324-9256